

3rd IMA Conference on the Mathematical Challenges of Big Data

Oral Presentation

GABRIELE LA MALFA

MSc Quantitative Finance

11/12/2018

Title:

Time Series Forecasting and Anomaly Detection using Machine Learning Algorithms: a Comparison of Performances

Topic:

Time Series Forecasting and Anomaly Detection using Sequentially Discounting Autoregressive algorithm (SDAR) and Long Short-Term memory Recurrent Neural Network (LSTM RNN algorithm)

Keywords:

Time Series Analysis, Machine Learning, Outliers Detection, Forecasting, Simulation

SDAR Algorithm: Sequentially Discounting AR Model

SDAR algorithm is a statistical based algorithm, which permits to learn a statistical behaviour of data provided.

Data behaviour is called "statistical regularity". Once the statistical behaviour has been learnt, data can be compared to it and therefore identified through different criteria such as outliers or not.

SDAR Algorithm: Characteristics

The process implemented is an online algorithm. This allows to immediately determine whether or not a given subset of points represents outliers.

Another feature of the model is the ability to detect anomalies even in the presence of a non-stationary process (stationarity is often considered optimal for working with other models)

SDAR model as presented in [1], assumes a Gaussian Mixture as a statistical model for continuous variables.

The AR algorithm permits to update the parameters of the Gaussian distribution, applying a discounting effect on past data.

SDAR Algorithm: Characteristics

Given that the Probability Density Function is a Multivariate Gaussian function as follows:

$$p(x_t | x_{t-k}^{t-1} : \theta) = \frac{1}{\sqrt{(2\pi)^{k/2} |\Sigma|}} \exp\left(-\frac{1}{2}(x_t - \omega)^T \Sigma^{-1} (x_t - \omega)\right) \quad (1)$$

\hat{x}_t and $\hat{\Sigma}$ are the parameters of the Multivariate Gaussian.

$$\hat{x}_t = \hat{\omega}(x_{t-k}^{t-1} - \hat{\mu}) + \hat{\mu} \quad (2)$$

$$\hat{\Sigma} = (1 - r)\hat{\Sigma} + r(x_t - \hat{x}_t)(x_t - \hat{x}_t)^T \quad (3)$$

\hat{x}_t is the weighted distance of each $x_t \in k$ from the updated mean $\hat{\mu}$, plus the updated mean $\hat{\mu}$.
 $\hat{\mu}$ is updated as follows:

$$\hat{\mu} = (1 - r)\hat{\mu} + rx_t \quad (4)$$

SDAR Algorithm: Characteristics

Covariance function (C) is a weighted sum of the previous covariances between steps belonging to k and ω_i (*weights* variable in the implementation) are the solutions to equation (5) and they are used to compute \hat{x}_t in equation (2).

$$C_j = \sum_{i=1}^k \omega_i C_{j-i} \quad (j = 1, \dots, k) \quad (5)$$

Using a loop *for j in range (k)*, C_j is computed as follows:

$$C_j = (1 - r)C_j + r(x_t - \hat{\mu})(x_{t-j} - \hat{\mu})^T \quad (6)$$

Scoring: Outliers and Change Points Detection

For outliers detection, the following algorithm is proposed:

$$Score(x_t) = -\log p_{t-1}(x_t), \quad (7)$$

which is the loss (logarithmic loss) performed regarding the point x_t , using a prediction based on the probability density at time $t - 1$.

The following formula computes the change point scoring:

$$Scoring(T) = 1/T \sum_{i=t-T+1}^t score(x_i) \quad (8)$$

Through algorithm (8), a time series of averaged scores is obtained. This time series is used in SDAR to calculate new distribution parameters (x_{hat} and σ_{hat}) and a sequence of pdf denoted as q_t with $t = 1, 2 \dots t$.

When T_1 is set as the new window constant, the new score is calculated as follows:

$$Score(t) = 1/T' \sum_{i=t-T'+1}^t (-\ln q_{i-1}(y_i)) \quad (9)$$

Test: Data

To perform test section, two dataset of the same index are used:

TOPIX year: 1985-1995

FTSEurofirst 300 year: 2007-2018

TOPIX and FTSEEurofirst 300 data are taken from S&P Capital IQ and they are normalized according to min-max normalization.

Change point detection is performed using formulae (8) and (9).

SDAR parameters are: $r = 0.005$ and $k = 15$.

For change point detection, $T = 5$ and $T_1 = 5$.

Figure 1: TOPIX



Figure 2: FTSEEurofirst 300



LSTM RNN: Characteristics

Recurrent Neural Networks (RNNs) are a sequence of Neural Networks (NNs) N_1, N_2, \dots, N_n arranged each at every single step of a discrete process.

Recurrent Neural Networks are able to take a sequence of input-values x_1, \dots, x_n one at a time leading the information along the process.

Inputs are processed by the neural networks and a vector of parameters (weights) is updated. These parameters contain the information coming from the past Neural Networks: in this way the information flows through the Neural Networks.

LSTM RNN: Forward Propagation

RNNs forward propagation process is formalized as follows:

$$a_t = f(W_{aa}a_{t-1} + W_{ax}x_t + b_a) \quad (10)$$

$$\hat{y}_t = g(W_{ya}a_t + b_y) \quad (11)$$

Where: a is the activation function of the next NN;

W_{aa} , W_{ax} , W_{ya} are respectively the weights related to the next activation function, the NN inputs and the NN output;

x_t is the input of the NN at step t ;

$f()$, $g()$ are a non-linear function (in literature hyperbolic tangent and softmax are respectively used as f and g function);

\hat{y}_t is the output of the NN at step t ;

b_a and b_y are the biases of activation functions and outputs

LSTM RNN: Loss

After the computations of the forward propagation, it is necessary to calculate the Loss of the model which is translated in the summation of the losses of the model for each training step and each pair of values of the sequence $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$.

The single-step loss is: $L_t(\hat{y}_t, y_t)$.

The overall loss is: $L(\hat{y}, y) = \sum_{t=1}^t L_t$.

The loss of the model is calculated in accordance with the intended purpose. Generally, the loss is calculated as the "distance" between the prediction \hat{y} coming from the networks' computations and the label of the supervised model y .

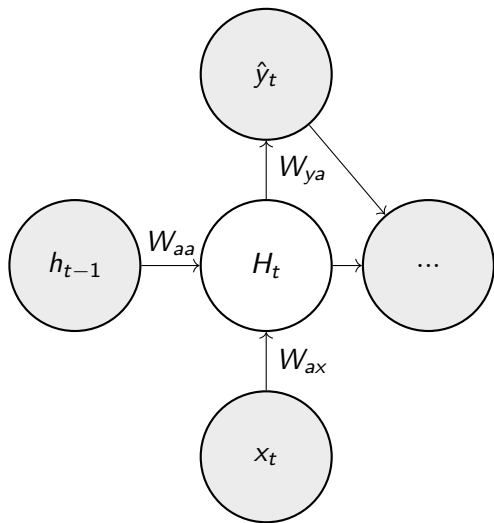
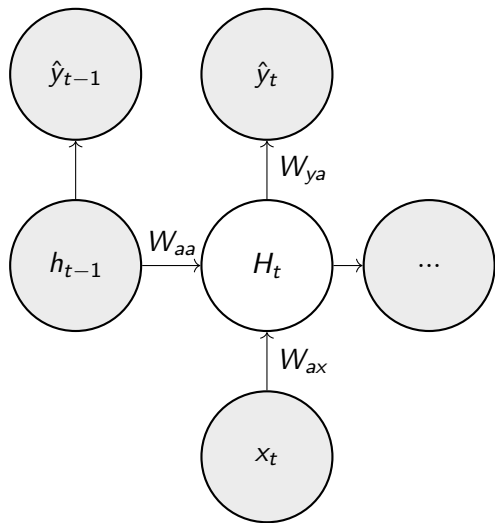
LSTM RNN: Information and Backpropagation

There may be different architectures for the RNNs: for example the hidden layer h_t could depend on the hidden layer h_{t-1} at the previous step or imply in the calculation also the output y_{t-1} of the previous step. The different architecture defines the passage of information between one RNN cell and another.

In the deep neural networks, even if the weights are shared, the way in which the passage of the information happens turns out to be very difficult to be interpreted: the parameters to be updated are many and their change often makes lose of interpretability so much that often the neural networks are used as black boxes.

Backpropagation is used to update the parameters of the RNN. It is often based on an optimization process, calculating the partial derivatives of the Loss with respect to the weights (W_{aa} , W_{ax} , W_{ya}), calculating the gradient of the loss function and finally using an optimization method to minimize the loss function.

LSTM RNN: Different RNN Architectures



LSTM RNN: High Temporal Distance Relationship Learning

Parameter updatings depend on how the loss is optimized and how the gradient converges to the minimum in backpropagation: weights update is proportional to the partial derivatives of the loss function.

There may be two pathological situations:

- Vanishing gradients: if the weights are small, the gradient convergence to the minimum may be very small or stop.

- Exploding gradients: if the weights are large, the learning mechanism may diverge.

LSTM networks implement a system of gates (Input Gate, Output Gate and Forget Gate) leading the activation or the inhibition of the information in the cell memory:

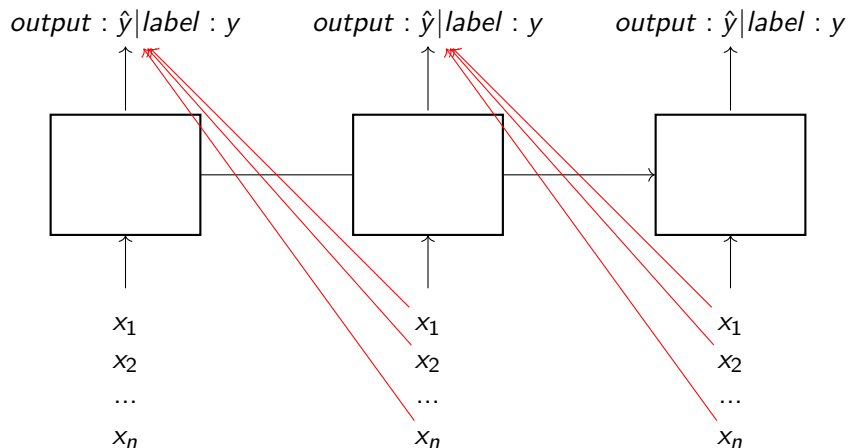
- Input Gate: the memory cell retains the previous state.

- Forget Gate: resets the state of the cell.

- Output Gate: it enables the information to be carried on or not.

LSTM model permits to manage the information and the overall recurrent process for a long time series.

LSTM: Architecture



The basic idea is that the RNN (LSTM) learns the weights using the next step input x_{t+1} as the label of the previous step t . The error is the difference between the prediction \hat{y}_t and the label y (next step input x_{t+1}).

Test: Data

To perform test section, three datasets are used:

TOPIX year: 1985-1995

FTSEurofirst 300 year: 2007-2018

ABSKF (FTSEurofirst 300) year: 2007-2018

The whole historical series relative to the three datasets, is subdivided in train, validation and test according to different percentages established by the experiments.

Anomaly Detection with LSTM

The following procedure is used to do anomaly detection with the LSTM model:

1. Training of the model on train part of the dataset;
2. Error computation on validation part as the difference between the label and the prediction;
3. Assuming that the errors are distributed according to a Gaussian distribution, the parameters μ and θ are fitted on validation part with MLE;
4. With new data coming from the test part, once the test errors are estimated, we calculate the probability $p(t)$ of observing each error according to the previous Gaussian;
5. If $p(t)$ is lower than a fixed threshold τ , then we are in presence of an anomaly.

Figure 3: TOPIX

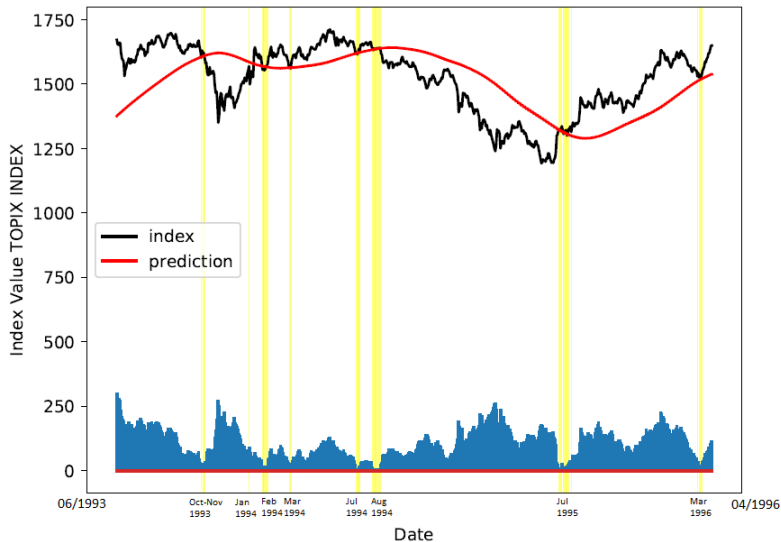


Figure: Change Point Detection

Figure 4: FTSEurofirst 300

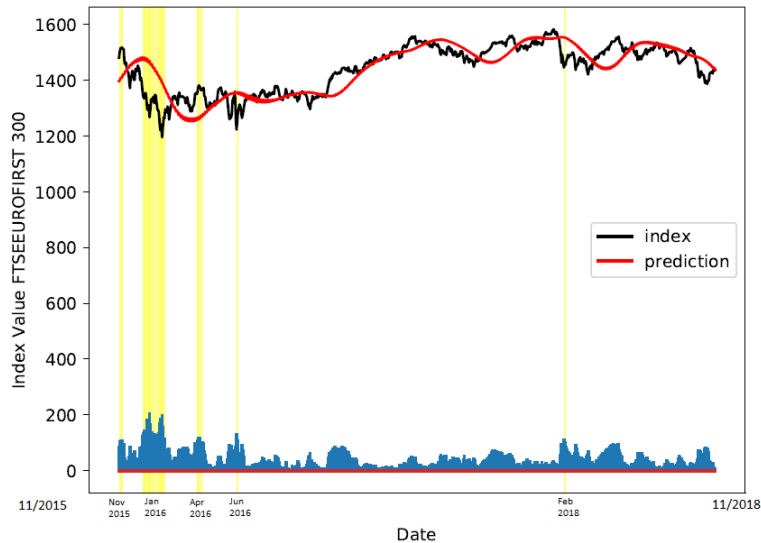


Figure: Change Point Detection

Figure 5: ABSKF

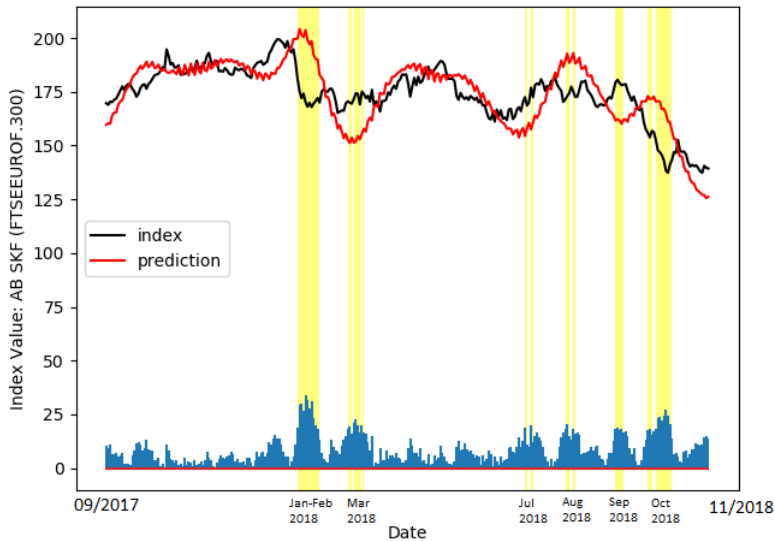




Figure: Change Point Detection

A special thanks to my brother Emanuele for his contributions and his support so that this first presentation of mine can be presented here

Thank you for your attention

Bibliography

-  Takeuchi, J, Yamanishi, K, A unifying framework for detecting outliers and change points from time series, *IEEE Transactions on Knowledge and Data Engineering*, Volume 18, Issue 4, April 2006.
-  Malhotra,P, Vig, L, Shroff, G, Agarwal, P, Long Short Term Memory Networks for Anomaly Detection in Time Series, *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, Bruges (Belgium) 22-24 April 2015.